# Data Visualization with Python

Pawel Rosikiewicz 2019.05.20

## INTRODUCTION

### Exploratory Data Analysis (EDA)

**1. To organize and summarize the data**

- to understand underlying **structure of the data**
- to Identify & measure the **abundance & distributions of missing data**
- to Identify **outliers**
- to refine **feature selection**
- to discover **meaningful features and patterns**
- to create **composed features**, if needed

**2. To prepare Graphical Summaries**

- quickly visualize the data
- investigate relationships between different variables of the data

**3. To perform pre-hypothesis testing**

- to validate assumptions that might have been done about the data, eg: random sampling, no differences between batches, etc…

## DATA TYPES

**Quantitative;**

- Continuous
- Discrete

**Categorical**;

- Ordinal - ordered categories, eg: small, intermediate, big,
- Nominal - unordered categories, eg: gender, nationality, etc...

## POPULAR PACKAGES

**Matplotlib**

- **Import matplotlib as plt**
- https://matplotlib.org/index.html
- Plots are created one step at a time! We first create a figure with its axes, and then fill the axes with information such as, data points, labels, color. In many cases, more than one plot can be placed on one fig.

**Seaborn;** statistical data visualization

- **Import seaborn as sns**
- https://seaborn.pydata.org
- Built on top of Matplotlib, **has sophisticated** styles and color palettes. Some of its figures, like boxplot, or histogram are a plot. That, means we can, not use plt functions to modify them (eg to change their size, or plot with other figures on the same plot)

**ggplot**

- **from ggplot import ***
- https://yhat.github.io/ggpy/
- Ggplot operates differently compared to Matplotlib:. it lets users layer components to create a full plot. For example, the user can start with axes, and then add points, then a line, a trend line, et
- Allows using high-level grammar based on R's ggplot2 and the Grammar of Graphics link: https://www.amazon.com/Grammar-Graphics-Statistics-Computing/dp/0387245448
- ggplot examples in R: http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html

**bokeh**

- https://bokeh.pydata.org/en/latest/docs/user_guide/plotting.html
- as ggplot, its syntax is based on Grammar of Graphics
- **Web compatible:** supports streaming, and real-time data and its unique selling proposition is **its ability to create interactive, web-ready plots**, which can easily output as JSON objects, HTML documents, or interactive web applications.
- Three interfaces with varying degrees of control:

  **Top most level** for creating charts quickly. Methods for creating common charts such as bar plots, box plots, and hist.

  **The middle level,** allows the user to control the basic building blocks of each chart (eg: dots in a scatter plot)

  **The bottom level;** for developers and software eng.; no pre-set defaults and requires to define every element of the chart.

**Plotly**

- **import plotly.plotly as py**
- https://plot.ly
- **online platform for data visualization**; allows producing interactive plots, and it offers some charts not found in most libraries, like contour plots https://plot.ly/python/contour-plots/ used for SÊRRO visualization in ML algorithms (gradient descent)
- can be accessed from Python notebook

**Pygal**

- http://pygal.org/en/stable/
- **Interactive plots that can be embedded in a web browser**
- good build in styles.
- Scalable Vector Graphics (SVG) plot format for web; XML-based vector image format for 2D graphics with support for interactivity and animation. SVG is very good for small datasets, but with large datasets the chart will become sluggish and have trouble rendering

**Gleam package**

- inspired by R's Shiny package. It **allows to turn any analysis into interactive web apps using only Python scripts**. Gleam users don't need to know HTML, CSS, or JavaScript to do this.
- Works with any Python data visualization library. Once users have created a plot, they can build fields on top of it to filter and sort data.

**Missingno;**

- **Missing data visualization module for Python**, The user can filter and sort data based on completion or spot correlations with a heat map or a dendrogram.
- from GFitHub: https://github.com/ResidentMario/missingno
- https://www.residentmar.io/2016/03/28/missingno.html

**Geoplotlib**

- for plotting geographical data and map creation
- used to create a variety of map-types, like choropleths, heatmaps, and dot density maps. Pyglet (an object-oriented programming interface) is required to be installed in order to use Geoplotlib. A package offers the set of in-built tools for the most common tasks such as density visualization, spatial graphs, and shape files

**Altair**

- Declarative Visualization in Python
- https://altair-viz.github.io
- To get examples, with code, just click on the plots you like on that page

## POPULAR FIGURE TYPES

**Showing potential relationships between two variables**

- **Scatter plots;** -||-
- **Line plots;** there can be only one y-value for each x-value.

**Presenting categorical data**

- **Bar charts**, to present the amount or proportions of categorical data
- **Pie charts;** numerical proportion of categorical variables
- **Stem plots;** all points connected with line to x or y axis

**For data distribution**

- **Histograms;** to evaluate the distribution of data
- **Density plot;** to visualises the distribution of data over a continuous interval or time period, with smoothing that removes noise.
- **Boxplot;** a standardized way of displaying the distribution of data based on a five-number summary

**For plotting multidimensional data (eg: x~y~z)**

- **Contour plots**; to show 3D structure on 2D arrea, like a map with lines showing the elevation, used in ML
- **Quiver plots**; display vectors as arrows at the points
- **Spectrograms;** 3D on 2D space, 3$^{rd}$ dimension is a coloration
- **Heatmaps;** two categorical data for rows and cols, and categorical or numerical data in cells

# Data Visualization with Python

Pawel Rosikiewicz 2019.05.20

## POPULAR PLOT TYPES

## Part 1

### HISTOGRAMS

https://datavizcatalogue.com/methods/histogram.html

**For:** To determine information about the distribution of variables in a data set

- **is the distr. Unimodal** (one peak), **bimodal** (two peaks), or **uniform**
- … **symmetric** or asymmetric
- … **skewed**, if yes, is it skewed left or right
- are they **outliers**
- what is the **span**, or the difference between the min and max values

**How many bins should be used in histograms?**

Use square root of the number of points in the data set as nr of bins.

```
> import numpy as np
  import math
  bin_number_for_hist = np.floor(math.sqrt(1000))
```

### BOXPLOTS

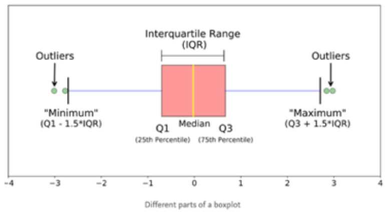
https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

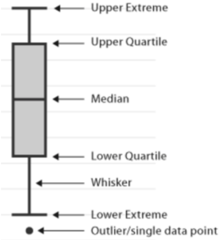
**Def:** A standardized way to displaying data distribution through their quartiles

**For:** Provides information on the variability or dispersion of the data:

- Are they **outliers in each dataset** and what are their values?
- Are your data **symmetrical**?
- How tightly your data are **grouped**?
- How your data is **skewed**?

| | |
|---|---|
| **Median** (Q2/50th Percentile) | the middle value of the dataset. |
| **First quartile** (Q1/25th Percentile) | the middle number between the smallest number (not the “minimum”) and the median of the dataset. |
| **Third quartile** (Q3/75th Percentile) | the middle value between the median and the highest value (not the “maximum”) of the dataset |
| **Interquartile range** (IQR) | 25th to the 75th percentile. |
| **Whiskers** | The whiskers end at the highest point within $Q3 + 1.5R$ and at the lowest point above $Q1 - 1.5R$. So for instance if $Q3+1.5R=100$ and the highest value in your sample is 90 then the whisker will end at 90. All values >100 will be outliers! |
| **Outlier** | all values above **“maximum”**: Q3 + 1.5*IQR, or below the **“minimum”** Q1 - 1.5*IQR used for plotting wiskers. |





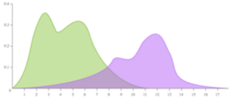
### DENSITY PLOT

https://datavizcatalogue.com/methods/density_plot.html

**Synonyms:** Kernel Density Plots, Density Trace Graph

**For:** DP visualize the distribution of data over a continuous interval or time period, a variation of a Histogram that uses kernel smoothing to plot values, (noise smoothing). The peaks of a Density Plot help display where values are concentrated over the interval.

**Advantage over the histogram:** DPs are better at determining the distribution shape because they're not affected by the number of bins used.



### SCATTER PLOT

https://datavizcatalogue.com/methods/scatterplot.html

**Synonyms:** Scatter Graph, Point Graph, X-Y Plot, Scatter Chart or Scattergram.

**Def:** Scatterplots use a collection of points placed using Cartesian Coordinates to display values from two variables.

**For:** Detecting relationship / **correlation** between the two numerical variables, different types of relationships can be detected:

- **Positive correlation** (values increase together),
- **Negative correlation** (one value decreases as the other increases),
- **Null** (no correlation),
- **linear, exponential c.**
- **U-shaped c.**

**Line of Best Fit** or a **Trend Line**; Lines or curves that drawn as close to all the points as possible

**Strength of the correlation**; shows how closely packed are the points to each other on the graph. Points that end up far outside the general cluster of points are known as **outliers**.

### LINE PLOT

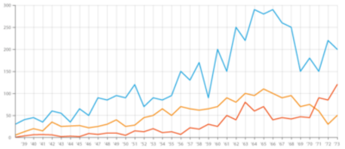
https://datavizcatalogue.com/methods/line_graph.html

**Def:** drawn by plotting data points on a Cartesian coordinate grid, then connecting a line between all of these points. Typically, the y-axis has a quantitative value, while the x-axis is a timescale or a sequence of intervals. Negative values can be displayed below the x-axis.

**For:** To display quantitative values over a continuous interval or time period. Often used to show **trends** and to analyze how the data has changed over time.

- **upward slope** indicates where values have increased
- **downward slope** indicates where values have decreased.

**Caution:** use no more than 3-4 lens per graph, or divide the chart into smaller multiples



### BAR CHART

https://datavizcatalogue.com/methods/bar_chart.html

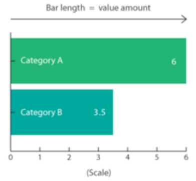
**Def:** A graph used to visualize categorical data with rectan-gular bars with heights (column chart, vertical bars) or lengths (horizontal bars) proportional to their values. Axis 1: categorical data that are being compared, axis 2: discrete value scale used to compare different categories.

**For:** Bar Chart's discrete data are categorical data and therefore it answers the question of "**how many?**" in each category.

**Not a histogram:** Unlike histograms, bar chart do not display continuous developments over an interval.

**Caution:** labelling becomes problematic with large number of bars.

**Multi-set Bar Charts:** Used to compare grouped variables or categories to other groups with these same variables or category types. Multi-set Bar Charts can also be used to compare mini Histograms to each other, so each bar in the group would represent the significant intervals of a variable.



# Data Visualization with Python

Pawel Rosikiewicz 2019.05.20

## POPULAR PLOT TYPES

## Part 2

### STACKED BAR GRAPH

https://datavizcatalogue.com/methods/stacked_bar_graph.html

**Def:** multiple datasets (groups) are represented as bars stacked on top of each other (segmented bars). Each bar represent one dataset, and each segment a category in different datasets. Twoi types (see below)

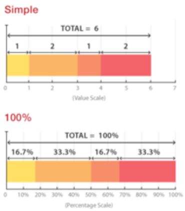
**For:**

- How a larger category is divided into smaller categories?
- What the relationship of each part has on the total amount?.
- What is the total amount of all elements in each group?

**Simple Stacked Bar Graphs** place each value for the segment after the previous one. The total value of the bar is all the segment values added together. **Used for** comparing the total amounts across each group/segmented bar.

**100% Stack Bar Graphs** show the percentage-of-the-whole of each group and are plotted by the percentage of each value to the total amount in each group. **Used for** visualization of differences in the relative quantities in each group.

**Caution:** comparing corrersponding segments in different groups is difficult, as they're not aligned on a common baseline.



### MOSSAIC PLOT

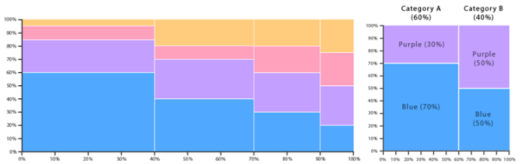
https://datavizcatalogue.com/methods/marimekko_chart.html

**Synonyms:** Marimekko Plot

**Def:** Both axes are variable with a percentage scale, that determines both the width and height of each segment (two-way 100% stacked bar chart)

**For:** to visualize categorical data over a pair of variables. to detect relationships between categories and their subcategories via the two axes.

**Caution:** These plots can be hard to read, especially when there are many segments, and to compare corresponding segments in different groups.



### HEATMAP

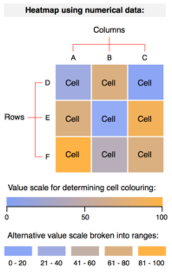
https://datavizcatalogue.com/methods/heatmap.html

**Def:** Heatmaps visualize numerical or categorical data through variations in coloring on the intersection of variables from two different categories (labels on rows and columns)

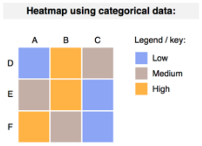
- **Categorical data;** colour-coded
- **Numerical data** requires a **colour scale,** a selection of solid colours can be used to represent multiple **value ranges** (0-10, 11-20, 21-30, etc) or you can use a **gradient scale** for a single range (for example 0 - 100) by blending two or more colours together.

**For:**

- **Typically used for generalized view over numerical data**: it is hard to accurately tell the differenc between colour shades and to extract specific data points from (unless, you include the raw data in the cells). Thus, heatmaps are better suited to visualise generalized view of numerical data
- **for cross-examining multivariate data stored in a tabular format (2D)** and for showing the variance across multiple variables, revealing any patterns, displaying whether any variables are similar to each other, and for detecting if any correlations exist in-between them.
- **to show the changes in data over time** if one of the rows or columns are set to time intervals. Eg: the temperature changes across the year in multiple cities (rows: the citie; columns: each month; cells: temperature values)



Heatmap using numerical data:
Columns
A
B
C
Rows
D
E
F
Cell
Cell
Cell
Cell
Cell
Cell
Cell
Cell
Cell
Value scale for determining cell colouring:
0
50
100
Alternative value scale broken into ranges:
0 - 20
21 - 40
41 - 60
61 - 80
81 - 100



### PIE CHART

https://datavizcatalogue.com/methods/pie_chart.html

**For:** To show proportions and percentages between categories, by dividing a circle into proportional segments. Ideal for giving the reader a quick idea of the proportional distribution of the data

**Problems are:**

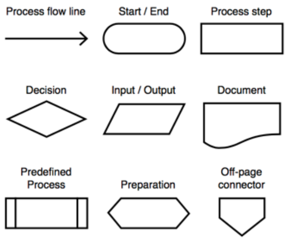
- No error bars
- unsuitable for large amounts of data with many categories
- take more space then their alternatives
- not good for making accurate comparisons between groups of Pie Charts (it is harder to distinguish the size of items via area when it is for length)

**Alternatives**

- 100% stacked bar chart
- Donut chart; easier to compare segments between charts
- Polar Area diagram (stacked bar plot on a polar coodinates)

### FLOW CHART SYMBOLS

https://datavizcatalogue.com/methods/flow_chart.html



Process flow line

Start / End

Process step

Decision

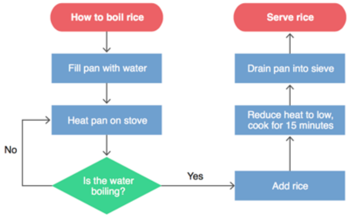
Input / Output

Document

Predefined Process

Preparation

Off-page connector



How to boil rice

Fill pan with water

Heat pan on stove

Is the water boiling?

No

Yes

Add rice

Reduce heat to low, cook for 15 minutes

Drain pan into sieve

Serve rice

# Data Visualization with Python

Pawel Rosikiewicz 2019.05.20

## ONLINE RESOURSES

### INTRESTING PAGES

**The data visualization catalogue;**

https://datavizcatalogue.com/index.html

excellent for learning and selecting figures by type and function. Blog has simple examples with definition of each graph type and keywords.Some graphs have links to examples in Python or R

**Python graph gallery with the code**

https://python-graph-gallery.com/all-charts/

**Titanic data plots in seaborn**

https://www.kaggle.com/fourbic/visualizing-the-titanic-data-with-seaborn

### DATA SET EXAMPLES

**SNS datasets**

https://github.com/mwaskom/seaborn-data

**sns.get_dataset_names();** to see all available datasets stored on github

['anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamonds', 'dots', 'exercise', 'flights', 'fmri', 'gammas','iris', 'mpg', 'planets', 'tips', 'titanic']

**sns.load_dataset( )** loads given data set as pandas df, using pandas.read_csv

```
import seaborn as sns
import pandas as pd
titanic = sns.load_dataset('titanic')
```

### DATA SET EXAMPLES

```
# data - iris

from sklearn import datasets

iris = datasets.load_iris() # Caution: Iris is sklearn.utils.Bunch object!!

                    # access by iris['target'] -> gives numpy.ndarray
```

```
# store the data as pandas df

iris_df = pd.DataFrame(iris['data'],

columns=iris['feature_names'])

iris_df['species'] = iris['target'] # extra column for further analysis

iris_df.head()
```

# Data Visualization with Python

Pawel Rosikiewicz 2019.05.20

## MATPLOTLIB - EXPLICIT WAY

https://matplotlib.org/tutorials/introductory/usage.html

### IMPORTS

**import matplotlib.pyplot as plt** ☺

**%matplotlib inline,** display all the plots directly inside the notebook, in cell below the code

Caution: all code must be in the same cell (Jupyter)

### GENERAL INFO

**Three levels of control**:

- * **plt** general functions, to create a figure
- * **fig** to control an entire figure, eg: style, size
- * **ax** methods allowing constructing each chart

**axes** the region of the image with the data space, one figure can have many axes, each axes contains 2 or 3 axis for 2d and 3d images respectively,

**axis** number-line-like objects, set the graph limits, generate the ticks and ticklabels.

**Input data**

- * **np.array** **np.array** or **np.ma.masked_array** always work
- * **pd.dataframe** can be used with some function eg:
  - ➢ ax[0].scatter(“x”, “y”, data=df) # where “x” and “y” are column names in dataframe df
- * **best way:** convert all array like obj’s to np.array objects.
  - ➢ a = pandas.DataFrame(np.random.rand(4,5), columns = list('abcde'))
    a_asarray = **a.values**
  - ➢ b = np.matrix([[1,2],[3,4]])
    b_asarray = **np.asarray( b )**

**Caution** if you use general functions such as plt.title, with an explicit way, they will be applied to the last plot, return an error, or generate an empty plot.

### PLT LEVEL

* **called before the plot**

**plt.figure()** > fig = plt.figure()
# an empty figure with no axes

**plt.subplots( )** always returns a list

**One plot** fig, ax = plt.subplots()
ax.plot(x, y) # axes sax with no indexes

**Multiple Plots** fig, ax = plt.subplots(nrow=2, ncol=2)
# creates a figure with a 2x2 grid of Axes
ax[0].plot(x, y)

**figsize = ( )** # figure size, by default in cm, so:

```
>    def cm2inch(value):  return value/2.54
     fig, ax = plt.subplots(nrows=1, ncols=1,
          figsize=(cm2inch(12), cm2inch(12)))
```

* **called after the plot**

**plt.legend( )** plt.legend( loc = "upper right" )

**plt.tight_layout( )** the plots will not overlap each other !

**plt.show( );** pushes the plot to be shown, add “;” at the end, to remove text messages form some functions, these are not errors

### FIG. LEVEL

* **called before the plot**

**fig.set_size_inches( )** fig.set_size_inches(12, 3)
# row length, col length, in inches

### AX. LEVEL

* **ax** list with axes, for each plot on a given fig. filled by row, eg: ax[3]: bottom left plot on 2x2 fig.

* **plots**

**ax[ ].plot(x, y)** **Line plot or dotted line plots**
x, y; 1d arrays,
> ax.plot( x, y, **color**='red', **linewidth**=2.0)
> ax.plot("col_name1", “col_name_2“, **data**=df )

**[fmt]** '[color][marker][line]` Each is optional: If line is given, but no marker, the data will be a line without markers eg:
plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)

**ax[ ].bar( )** **barplot**
x, y, **color** (can be in list), **edgecolor** (-||-)
**bottom** (where to start a bar, for stacked bar plots)

**ax[ ].scatter( )** **scatter plot**
**s** (size >0), **marker** eg: “*”; “.”, **color**,
**label** (name that will be displayed on legend
> ax.scatter(“x”, ” y”, s = 12, data=df,
label=list_with_labels, color= “red”,
marker="o")

* **plot description**

**ax[1].set_title( )** **fontsize, color, rotation**

* **x/y axis**

**ax.xaxis.set_visible( )** True/False, if F, no axix displayed on plot

**ax[1].set_ylim( )** tuple with 2 numbers eg: (0, 20)

**ax[1].set_ylabel( )** **fontsize, color, rotation**, input = str’s in list,

**ax[1].set_xticks( )** input = 1D np.array with numerical values or all expected ticks

**ax[1].set_xticklabels( )** input = list with strings, used as tick names, **fontsize, color, rotation**

### ATTRIBUTES

* **strings with text to display**

**$** obliquie letters; ax[1].set_title( “$***oblique***$”)

**“ ”.join( )** ‘_’.join( [ list wiht items to join ] )

**"\n"** new line

* **numerical values**

**.flatten()** use with all np.ndarrays, to ensure these are 1d, eg: ax.set_yticks(np.arange(10).flatten())

**np.linspace( );** (from, to, n); generate n numbers from to in equal dist. to included! # the default n is 50!

* **Colors:**

| | | | |
|---|---|---|---|
| 'b' | blue; | 'g' | green; |
| 'y' | yellow; | 'r' | red; |
| 'm' | magenta; | 'c' | cyan; |
| 'k' | black; | 'w' | white |

* **Markers**

| | | | |
|---|---|---|---|
| '.' | point marker; | 'o' | circle marker; |
| 's' | square marker | | |

* **Line:**

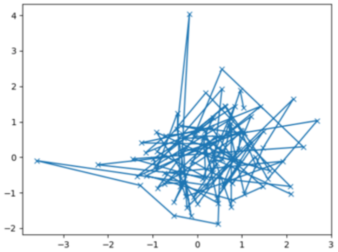
| | |
|---|---|
| '-' solid line style; | '--' dashed line style; |
| '-.' dash-dot line style; | ':' dotted line style |

### LEGEND

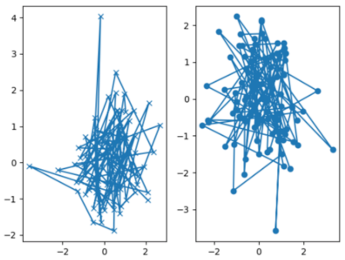
https://matplotlib.org/api/_as_gen/matplotlib.pyplot.legend.html

Function to package ploits, for making the same fgures with many different datasets. The recommended function signature is

```
def my_plotter(ax, data1, data2, param_dict):
    """
    A helper function to make a graph
    Parameters
    ----------
    ax : Axes
        The axes to draw to

    data1 : array
       The x data

    data2 : array
       The y data

    param_dict : dict
       Dictionary of kwargs to pass to ax.plot

    Returns
    -------
    out : list
        list of artists added
    """
    out = ax.plot(data1, data2, **param_dict)
    return out

# which you would then use as:
data1, data2, data3, data4 = np.random.randn(4, 100)
fig, ax = plt.subplots(1, 1)
my_plotter(ax, data1, data2, {'marker': 'x'})
```



```
fig, (ax1, ax2) = plt.subplots(1, 2)
my_plotter(ax1, data1, data2, {'marker': 'x'})
my_plotter(ax2, data3, data4, {'marker': 'o'})
```


4
3
2
1
0
−1
−2
−2
0
2
2
1
0
−1
−2
−3
−2
0
2